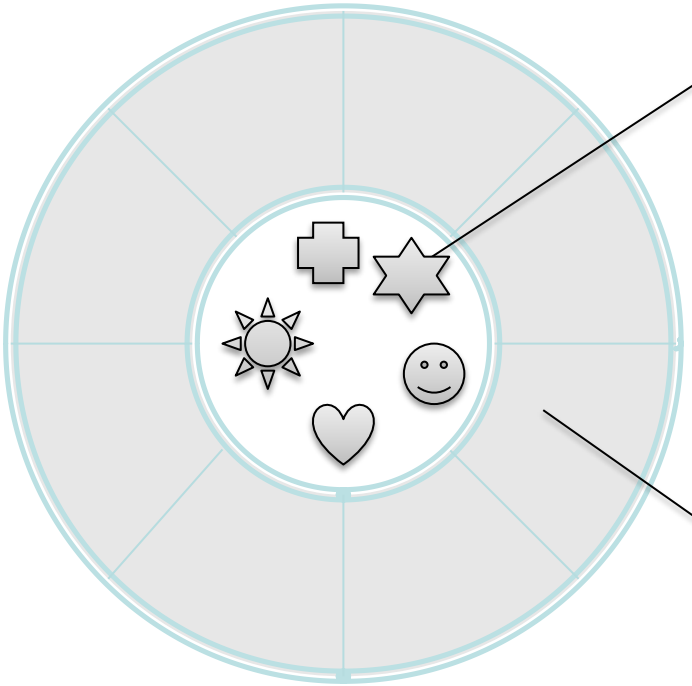
LE CLASSI

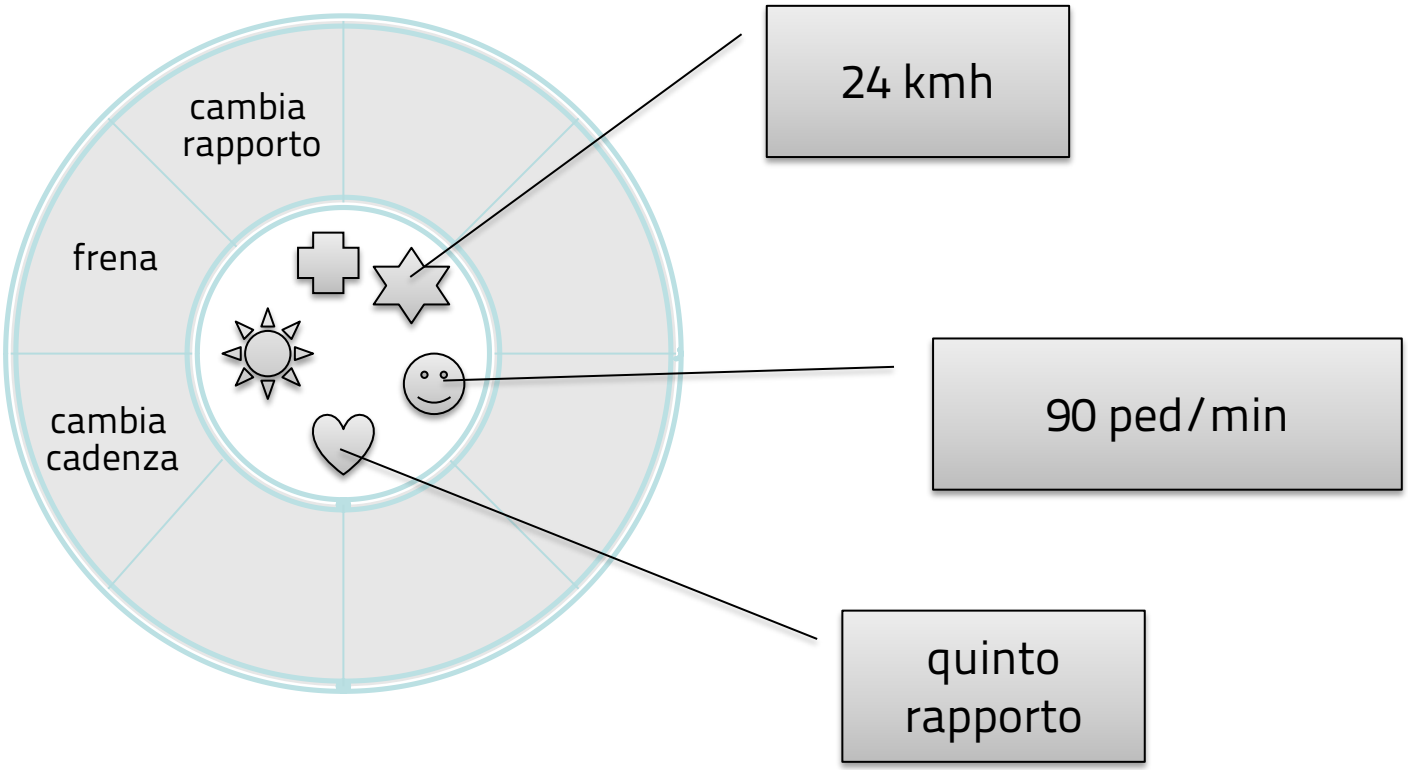
OGGETTO SOFTWARE



stati
o proprietà
(variabili)

comportamenti
o metodi
(funzioni)

BICICLETTA



ISTANZE E MEMBRI DI CLASSE

- Nel mondo reale, spesso troverete molti singoli oggetti tutti dello stesso tipo. Ci possono essere migliaia di biciclette esistenti, tutti della stessa marca e modello.
- Ogni bicicletta è stata costruita sualla base dello stesso progetto e quindi contiene gli stessi componenti.
- Una classe definisce un modello per un tipo di oggetto. Tutte le caratteristiche e i comportamenti che appartengono a una classe sono detti *membri* di tale classe. In particolare, le caratteristiche (nell'esempio del gatto, il nome, l'età e il colore) sono dette *proprietà* della classe e sono rappresentate da variabili, mentre i comportamenti (mangiare, dormire) sono detti *metodi* della classe e sono rappresentati da funzioni.

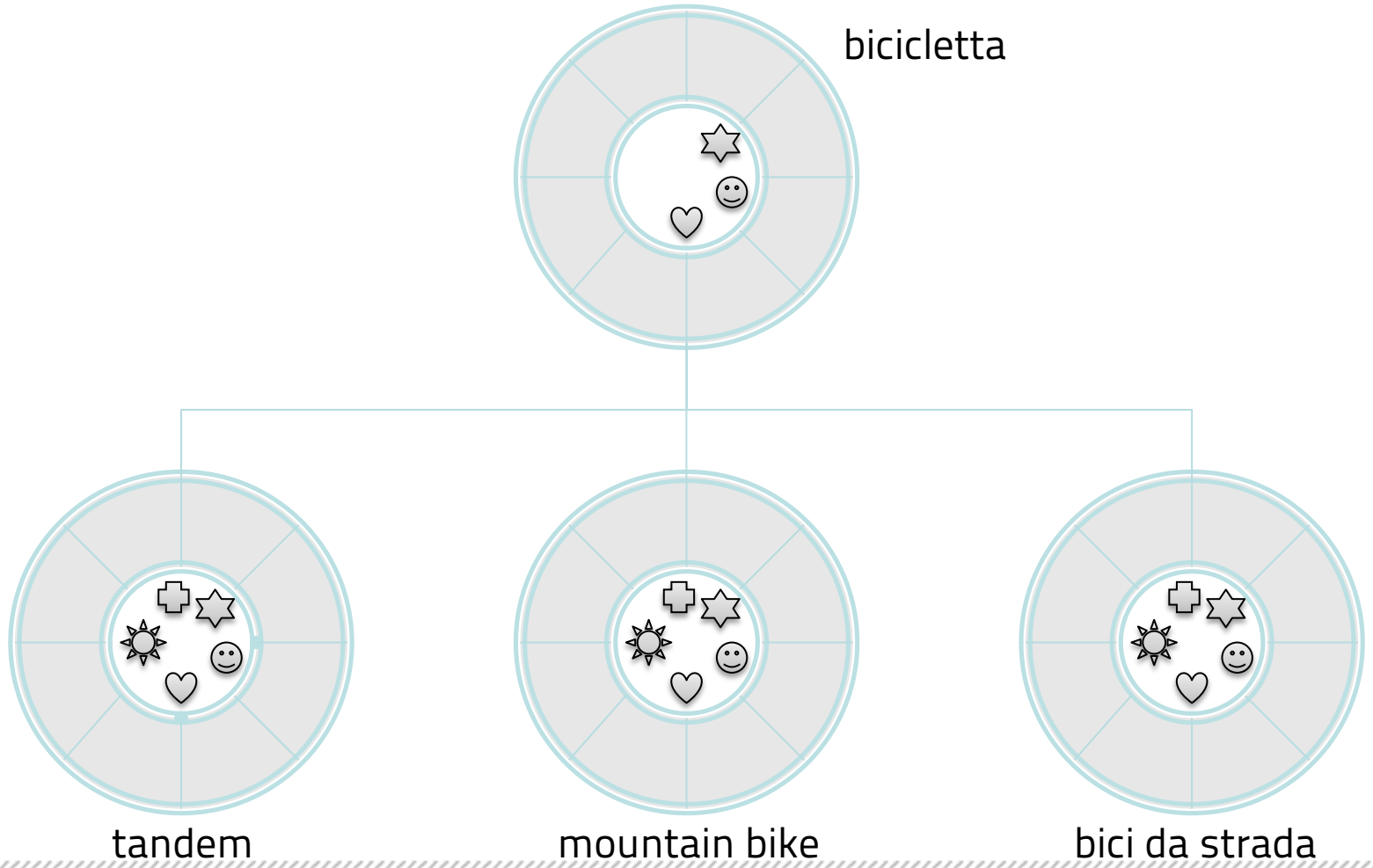
EREDITARIETÀ

- Mountain bike, biciclette da strada e tandem, condividono alcune caratteristiche (velocità, cadenza della pedalata, rapporto).
- Ma ciascuno definisce anche elementi aggiuntivi che le rendono differenti:
  - i tandem hanno due posti e due set di manubri,
  - le bici da strada hanno manubrio da corsa;
  - le mountain-bikes hanno una corona aggiuntiva che consente loro rapporti di trasmissione inferiori.
- La programmazione orientata agli oggetti consente di definire classi che *ereditano* gli stati e i comportamenti di altre classi. In questo esempio, bicicletta può essere una *superclasse* e MountainBike, Bici da strada, e Tandem delle sottoclassi.

EREDITARIETÀ

- Uno dei vantaggi principali della programmazione orientata agli oggetti è rappresentato dalla creazione di **sottoclassi** (tramite estensione di una classe); una **sottoclasse** eredita tutte le proprietà e i metodi della rispettiva classe.
- La sottoclasse definisce generalmente ulteriori metodi e proprietà o sostituisce metodi o proprietà definiti nella superclasse.
- Le sottoclassi possono inoltre sostituire i metodi definiti in una superclasse, ovvero fornire definizioni proprie per tali metodi.

EREDITARIETÀ



INTERFACCE

- Gli oggetti definiscono la loro interazione con il mondo esterno attraverso i metodi che espongono.
- I metodi costituiscono l'interfaccia con il mondo esterno, nello stesso modo in cui i pulsanti sulla parte anteriore del televisore, per esempio, sono l'interfaccia tra l'utente e l'elettronica che contiene involucro di plastica.
- Nella nostra classe bicicletta, l'interfaccia potrebbe essere per esempio l'elenco dei metodi che devo definire obbligatoriamente per creare una sottoclasse di bicicletta.

PACKAGE

- Un pacchetto è un namespace che organizza una serie di classi e interfacce correlate. Si può pensare di pacchetti come a diverse cartelle del computer. Poiché il software scritto in linguaggio ActionScript può essere composto da centinaia di singole classi è importante mantenere le classi organizzate in pacchetti.
- ActionScript fornisce una biblioteca (un insieme di pacchetti) che contengono le classi con cui abbiamo bisogna per gestire gli elementi. Questa libreria è noto come "Application Programming Interface", o "API".

CLASSI E TIPI

- Quando abbiamo parlato dei tipi di dati abbiamo visti che esistono *tipi di dati primitivi* e *tipi di dati derivati o complessi*.
- In un linguaggio orientato agli oggetto tutti i tipi di dati sono classi. Definiscono cioè non solo come un certo tipo di dato viene memorizzato nel computer, ma anche i metodi con cui posso interagire con esso.
- Creando una nuova classe il programmatore crea un nuovo *tipo di dati complesso*, un *modello*, cioè, di come posso gestire un determinato insieme di informazioni e di come posso interagire con esso.

LE CLASSI IN ACTION SCRIPT

LE CLASSI INCORPORATE

- Abbiamo visto che nei linguaggi OOP le classi servono a rappresentare *tipi di dati complessi*.
- Per gestire dati come le date, gli array, ecc Action Script fornisce un certo numero di classi incorporate nel linguaggio.
- Prima di passare alla programmazione di classi personalizzate vedremo di familiarizzare con il concetto di classe imparando ad utilizzare le classi incorporate in ActionScript.

LE CLASSI INCORPORATE

- *ActionScript* comprende numerose classi incorporate che servono a gestire diversi tipi di elementi: tipi di dati di base quali Array, Boolean, Date, gestione degli errori, gestione degli eventi, caricamento di contenuto esterno (XML, immagini, dati binari originari, ecc.).

LE CLASSI INCORPORATE

- Prima di passare all'uso delle classi è bene citare alcune regole di scrittura che ActionScript segue e alle quali, anche se non è sempre obbligatorio è bene adeguarsi:
  - i nomi delle variabili, delle proprietà, delle funzioni e dei metodi iniziano sempre con una lettera minuscola o, a volte, con '_" (underscore),
  - i nomi dei tipi di dati e, quindi, delle classi sempre con una lettera maiuscola.

COME USARE UNA CLASSE

- Nella maggior parte dei casi utilizzare una classe significa
  - creare un'istanza della classe stessa,
  - assegnare l'istanza creata ad una variabile e
  - usarne le proprietà e applicarne i metodi per ottenere i risultati desiderati.
- Per creare un'istanza di una classe devo applicare l'operatore ***new*** al ***costructor***, cioè alla funzione di costruzione della classe, una speciale funzione che ha lo stesso nome della classe.

ESEMPIO

```
// dichiaro “now” come variabile di tipo Date
var now:Date;
// assegno a now una istanza di Date che corrisponde alla
// data e all’ora corrente
now = new Date();
```

Dopo questa operazione now conterrà un’istanza della classe Date che rappresenta la data e l’ora corrente. A now si potranno applicare tutti i metodi e usare le proprietà della classe Date.

Da notare l’uso diverso di *Date* in *var now:Date;* e in *now = new Date();*. Nel primo caso *Date* indica il *tipo Date* e il costrutto dice al compilatore che la variabile now potrà contenere solo oggetti di tipo Date. Nel secondo caso **Date()** (seguito da parentesi) indica la funzione di costruzione della classe Date e il costrutto *now = new Date()* crea una nuova istanza della classe Date() e la memorizza in now.

ECCEZIONI

- L'operatore *new* non deve essere usato per i tipi primitivi: *Number*, *String*, *Boolean* (che comunque sono classi a tutti gli effetti, con le loro proprietà e i loro metodi). Le istanze delle classi sono create automaticamente quando maneggio questo tipo di dati.
- Ci sono delle classi per cui non si possono creare istanze e che rappresentano in quanto tali un oggetto su cui operare: *Accessibility*, *Camera*, *ContextMenu*, *CustmActions*, *Key*, *Math*, *Microphone*, *Mouse*, *Selection*, *SharedObject*, *Stage*, *System*. In questi casi proprietà e metodi si applicano direttamente alla classe e vengono detti *statici*.

LA CLASSE OBJECT

OBJECT

- Il tipo di dati *Object* rappresenta l'astrazione dell'idea di oggetto. Sta alla radice della gerarchia delle classi: tutte le classi sono estensioni di *Object*.
- La classe *Object* mi consente di creare oggetti vuoti, senza proprietà né metodi.
- Creando un'istanza di *Object* creo un oggetto personalizzato, a cui posso aggiungere le proprietà che voglio e che mi può servire per organizzare le informazioni nell'applicazione Flash.

ESEMPIO

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

Viene creato un nuovo oggetto denominato user e tre proprietà: name, age e phone che sono tipi di dati String e Numeric.
Lo stesso oggetto può essere creato anche assegnando alla variabile il letterale di tipo *Object* corrispondente.

```
var user:Object;
user = {name:”Irving”,age:32,phone:”555-1234”};
```

Quando si assegna ad una variabile un valore in formato letterale non è necessario richiamare il costruttore della classe con l’operatore *new*. Queato vale sia per *Object* che per *Array.*

for .. in

- Un ciclo **for..in** consente di eseguire iterazioni scorrendo gli elementi di un ogetto (un clip filmato, un oggetto generico, o un array).
- La struttura del comando è:

```
for variabile in oggetto
    blocco istruzioni;
```

- Il ciclo prende in esame tutti gli elementi presenti in <oggetto>. Ad ogni ciclo <variabile> assume il nome dell'elemento preso in esame e <blocco istruzioni> viene eseguito.

LA CLASSE ARRAY

ARRAY

- Un *array* è un oggetto le cui proprietà sono identificate da un numero (indice) che ne rappresenta la posizione nella struttura dell'array.
- Un array è un elenco di elementi recuperabile attraverso un indice.
- Non occorre che gli elementi dell'array abbiano lo stesso tipo di dati. È possibile inserire numeri, dati, stringhe, oggetti, array.

USO DEGLI ARRAY

- Gli array possono essere utilizzati in modi diversi.
- Uno degli utilizzi più tipici è quello di organizzare i dati di un database sotto forma di array di oggetti.
- La posizione di un elemento nell'array è detta *indice*. Tutti gli array sono con base zero, ovvero [0] è il primo elemento dell'array, [1] è il secondo, e così via.
- Normalmente i contenuti di un array vengono esaminati utilizzando un ciclo for che consente di scorrere tutti gli elementi di un array.

MODIFICA DI UN ARRAY

L'array può essere controllato e modificato tramite *ActionScript*. È possibile spostare valori all'interno dell'array o modificarne la dimensione. Il seguente codice, ad esempio, scambia i dati di due indici di un array:

```
var buildingArr:Array = new Array();
buildingArr[2] = "Accounting";
buildingArr[4] = "Engineering";
trace(buildingArr);
//undefined,undefined,Accounting,undefined,Engineering
var temp_item:String = buildingArr[2];
buildingArr[2] = buildingArr[4];
buildingArr[4] = temp_item;
trace(buildingArr);
//undefined,undefined,Engineering,undefined,Accounting
```

Nell'esempio precedente è necessario creare una variabile temporanea perché se il contenuto dell'indice 4 dell'array fosse stato copiato nell'indice 2 dell'array senza salvarne prima il contenuto, il contenuto originale dell'indice 2 sarebbe andato perso.

LUNGHEZZA DI UN ARRAY

- Quando si lavora con gli array, è spesso necessario conoscere il numero degli elementi contenuti in un array, in particolare se si scrivono cicli for che eseguono iterazioni su ogni elemento dell'array ed eseguono una serie di istruzioni. La proprietà *length* restituisce la lunghezza dell'array.

AGGIUNTA E RIMOZIONE

- Un array contiene elementi e ogni elemento ha una posizione numerica (l'indice) che corrisponde alla modalità con cui si fa riferimento alla posizione di ogni elemento nell'array.
- Ogni elemento può contenere dati o essere vuoto. I dati contenuti possono essere del formato numerico, stringa, booleano o essere un array o un oggetto.
- Se si assegna un solo valore in un array all'indice 5, la lunghezza dell'array restituisce 6. Nell'array vengono quindi inseriti cinque valori non definiti.
- Possono essere aggiunti elementi in coda all'array utilizzando il metodo *push().*

ARRAY ASSOCIATIVI

- Un array associativo è composto da chiavi e valori non ordinati e utilizza le chiavi alfanumeriche al posto degli indici numerici per organizzare i valori.
- Ogni chiave è una stringa univoca e viene utilizzata per accedere al valore a cui è associata. Il valore può essere un tipo di dati Number, Array, Object e così via.
- Array associativi e Object rappresentano due modi diversi per rappresentare gli stessi dati e sono intercambiabili.

```
// Definisce l'oggetto da utilizzare come array associativo
var someObj:Object = new Object();
// Definisce una serie di proprietà
someObj.myShape = "Rectangle";
someObj.myW = 480;
someObj.myH = 360;
someObj.myX = 100;
someObj.myY = 200;
someObj.myAlpha = 72;
someObj.myColor = 0xDFDFDF;
// Visualizza una proprietà utilizzando l'operatore punto e
// la sintassi di accesso agli array
trace(someObj.myAlpha); // 72
trace(someObj["myShape"]); // 72
```

LA CLASSE DATE

DATE

- La classe *Date* consente di recuperare i valori relativi alla data e all'ora del tempo universale (UTC) o del sistema operativo su cui è in esecuzione Flash Player.
- Per chiamare i metodi della classe Date, è prima necessario creare un oggetto Date mediante la funzione di costruzione della classe Date.

CREARE UN'ISTANZA DI DATE

- Posso passare alla funzione una data del passato o del futuro. Il questo caso Date() richiede da due a sette parametri (anno, mese, giorno, ora, minuti, secondi , millisecondi).
- In alternativa, posso costruire un oggetto *Date* passando un singolo parametro che rappresenta il numero di millisecondi trascorsi dal 1 gennaio 1970 alle 0:00:000 GMT.
- Se, infine, non specifico alcun parametro all'oggetto data Date() vengono assegnate la data e l'ora correnti.

USO DI DATE

- Una volta creato all'oggetto Date posso cozultare varie proprietà che:
  - Mi restituiscono informazione sull'anno, il mese, il giorno, il giorno della settimana, ecc della data creata.
  - Mi consentono di confrontare, sommare e sottrarre date

DATE

- È possibile recuperare singole informazioni utilizzando le proprietà di Date :
  - date - giorno del mese.
  - month - mese.
  - fullYear - anno.
  - day - giorno della settimana.
  - hours - ora.
  - minutes - minuti.
  - seconds - secondi.
  - milliseconds - millisecondi.